www.icgst.com

Minimum Average Scheduling Algorithm, MASA, Performance Boosting Approach

Kamal ElDahshan, Afaf Abd El-kader, Nermeen Ghazy
Dept. of mathematics, Computer science Division, Faculty of science, Al-Azhar University
Cairo, Egypt
[dahshan]@azhar.edu.eg, [afaa2islam, nermeen_ghazy89]@yahoo.com
http://www.azhar.edu.eg

Abstract
Scheduling is the process of allocating tasks to resources in order to optimize some objective function. There have been many algorithms used to schedule tasks on their resources .One of these scheduling algorithms is the Enhanced Max-min task scheduling algorithm which selects a task with average execution time(average or nearest greater than average) and assigns it to a corresponding resource producing minimum completion time. A drawback of this algorithm is that when some average is too large, it causes the makespan to increase. This paper proposes an efficient scheduling algorithm MASA (Minimum Average Scheduling Algorithm) which selects $\lfloor m/7 \rfloor$ (floor the number of resources divided by 7) tasks with minimum averages execution time (minimum averages or nearest greater than minimum averages) and then assigns these task(s) to the corresponding resources producing a minimum completion time. Experimental results ameliorate the makespan.

Keywords: *Scheduling, Max – min algorithm, Improved Max-Min Algorithm, Enhanced Max-min Task Scheduling Algorithm, Distributed system, Scheduling algorithm, Minimum Average Scheduling Algorithm.*

Nomenclature

M	number of resources
N	number of tasks
CPU	central processing unit
MASA	minimum average scheduling algorithm
T	task
R	resource
C_{ij}	completion time for task T_i on resource R_j
E_{ij}	execution time for task T_i on resource R_j

1. Introduction
Grid computing systems enable sharing large-scale resources among millions of computer systems such as the Internet. The grid infrastructure involves four levels. First: the foundation level, it includes the physical components. Second: the middleware level, it is actually the software responsible for resource management, task execution, task scheduling, and security. Third: the service level, it provides users with efficient services, Fourth: the application level, it contains the services such as operational utilities and business tools.
Scheduling has become one of the major research objectives, because it directly increases the performance of grid applications [1]. It manages jobs to allocate appropriate resources by using scheduling algorithms and policies [2].
Scheduling can be classified as either static or dynamic. In static scheduling, the information regarding all the resources as well as the tasks is assumed to be known in advance. Furthermore, each task is assigned once to a resource. On the other hand, in dynamic scheduling, the task allocation is done while the application executes where it is not possible to find execution times. Several heuristic algorithms for grid task scheduling have been developed to improve grid performance [3], [4].
The aim of the scheduling problem is to optimize some objective functions such as makespan, CPU utilization, throughput, turnaround, response time, waiting time or fairness [5].

- Makespan: the maximum completion time
- CPU utilization: keeping the CPU as busy as possible
- Throughput: the number of processes that are completed per unit of time
- Turnaround: the time between starting and completion
- Response time: the time from submission of a task to the first response

- Waiting time: the amount of time that is waiting in the ready queue
- Fairness: giving each process a fair share of the CPU

The main contribution of this work is to introduce an efficient algorithm for scheduling tasks on resources with minimizing the makespan.

The remainder of the paper is organized as follows: Section (2) focuses on some related work Section (3) focuses on Enhanced Max-min Task Scheduling Algorithm Section (4) presents the proposed algorithm MASA Section (5) describes Experimental data Section (6) explains Results analysis Section (7) concludes the paper and presents future work.

2. Related work

Many algorithms have been used to schedule tasks on their resources:

FCFS (First Come, First Served) algorithm: it is a non- preemptive algorithm. jobs are come on first executes and served first, it is so easy because it selects first process whatever has long or short execution time so the average waiting time is high which is poor in performance.

SJF (Shortest Job First) algorithm: Shortest Job First algorithm is a non – preemptive algorithm where in the ready queue, the short jobs are executed first. This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next execution time. If there is two processes have the same execution time, it is used FCFS scheduling.

SRTF (Shortest remaining time first) algorithm: it is a preemptive SJF algorithm; it preempts the currently executing process while a non-preemptive SJF algorithm will allow to finish the currently running process first.

Round Robin algorithm: Round Robin is the preemptive process scheduling algorithm. It has a quantum which is a fixed time provided for each process. While it is executed for given time period it is preempted and other process executes for given time period. It is used context switching to save states of preempted processes.

Priority scheduling algorithm: it is a non-preemptive algorithm. Each process is assigned a priority. The priority can be decided based on memory requirements, time requirements or any other resource requirement. The Process with highest priority will be executed first and so on. If the processes have the same priority, it will be executed on first come first served basis.

Multiple-Level Queues Scheduling: Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics. Each queue can have its own scheduling algorithm and priorities.

Min–Min algorithm: This algorithm finds the task which has a minimum execution time and assigns the task to the resource that produces minimum completion time. The ready time of the resource is updated. This procedure is repeatedly executed until all unmapped tasks are scheduled [6], [7], [8], [9].

Max–Min algorithm: The Max-Min scheduling algorithm schedules the larger task first. The ready time of the resource is then updated. This procedure is repeated until all-unscheduled tasks are assigned. If there is only one long task, the Max-min algorithm executes many short tasks concurrently with the long task and the makespan of the system is most likely determined by the execution time of the long task so Max-Min scheduling algorithm gives better results than Min–Min algorithm [6], [7], [8], [9].

RASA algorithm: this algorithm uses the Max-Min, Min-Min algorithms. If the number of available resources is odd, the Min-min algorithm is applied to assign the first task, whereas the number of available resources is even the Max-min algorithm is applied.
The remaining tasks are assigned to their appropriate resources by one of the two algorithms alternatively. If the Min-min algorithm is applied to assign the first task to resource, the next task will be assigned by the Max-min algorithm. In the next round the task assignment begins with an algorithm different from the last round. So if the first round begins with the Max-min algorithm, the second round will begin with the Min-min algorithm [8].

Improved Max-Min Algorithm: This algorithm uses the advantages of Max-Min and solves its disadvantages. It is concerned with the number of the resources and the tasks. It is based on the expected execution time instead of completion time. This algorithm calculates the expected completion time of the tasks on each resource. Then the task with the maximum expected execution time (Largest Task) is assigned to a resource that has the minimum overall completion time (Slowest Resource). Then, this scheduled task is removed from meta-tasks and all corresponding times are updated and then the traditional max-min algorithm is applied to the remaining tasks. The allocation of the slowest resource to longest task allows availability of high-speed resources for finishing other small tasks, which achieves the shortest makespan of submitted tasks on available resources [10].

Enhanced Max-min Task Scheduling Algorithm: This algorithm introduced a unique modification of Improved Max-min task scheduling algorithm. It is based on the

expected execution time instead of completion time. It assigns the task with average execution time (average or nearest greater than average Task) to the slowest resource produces minimum completion time. This reduces overall makespan and balance load across resources [11].

This algorithm is provably described below.

3. Enhanced Max-min Task Scheduling Algorithm

Enhanced Max-min algorithm is based on the expected execution time instead of completion time. In the set of tasks to be scheduled, the largest task may be too large compared to other tasks which cause increasing makespan. This occurs because first; the largest task is executed by the slowest resource that has the minimum overall completion time while other tasks are executed by faster resources. Hence, the Enhanced max-min algorithm selects a task with (average or nearest greater than average) execution time and assign it to the slowest resource produces minimum completion time. Then, this task is removed from meta-tasks and the all calculated times are updated. The traditional max-min algorithm is then applied to the remaining tasks [11].

4. The proposed algorithm: MASA (Minimum Average Scheduling Algorithm)

Sometimes in the matrix of execution time the average of some resource may be very large compared to other averages, applying the Enhanced Max-min algorithm will produce a great makespan. The proposed algorithm improves the Enhanced Max-min algorithm, instead of selecting average or nearest greater than an average task. MASA selects the $\lfloor m/7 \rfloor$ (floor the number of resources divided by 7)tasks that have (minimum average execution time or nearest greater than minimum average execution time). These tasks are then assigned to the corresponding resources and the traditional Max-min algorithm is applied. MASA decreases makespan because it is does not depend on the largest average and assigns minimum average tasks to faster resources which produce minimum completion time.

Minimum Average Scheduling Algorithm, MASA

The MASA algorithm is as follows

1: For all tasks t_i in Meta task
2: For all resources R_j
3: $C_{ij} = E_{ij} + r_j$
4: Compute average execution time for all resources
5: Select k= $\lfloor m/7 \rfloor$ task(s) with minimum averages
6: Assign selected task(s) to the corresponding resources R_j that gives minimum completion time
7: Delete selected task(s) from Meta task
8: Update completion times for corresponding resources
9: While there are tasks in Meta task
10: For each task find earliest completion time and the resource that obtains it
11: Find the task T_k with maximum earliest completion time
12: Assign task T_k to the corresponding resource R_j that gives minimum completion time
13: Update C_{ij} for all i
14: End while

The complexity of the proposed algorithm is $O(mn^2)$ as that of the Enhanced Max-min algorithm, where m is the number of resources in the system and n is the number of tasks which should be scheduled to be executed.

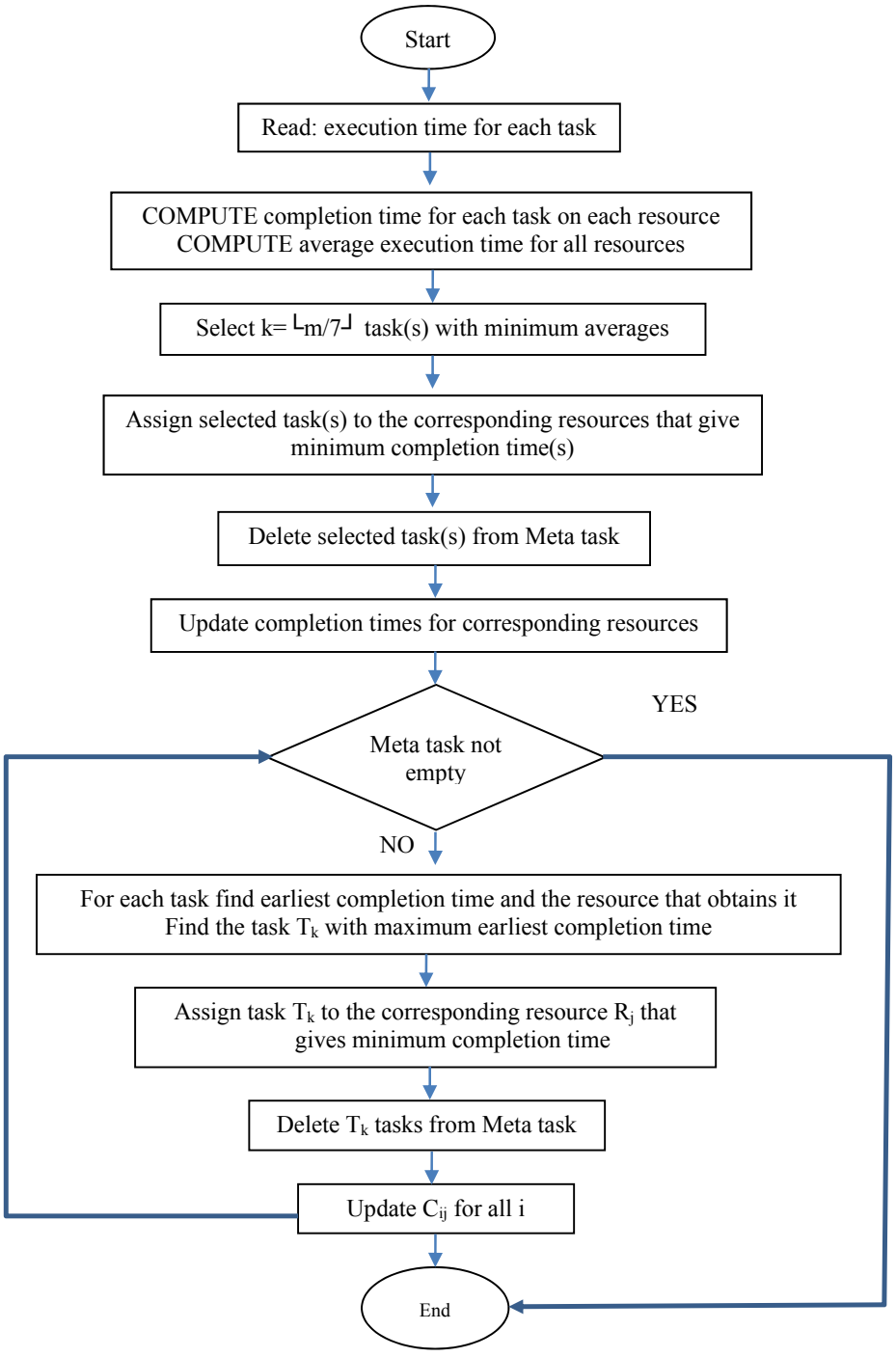
A Comparison between the Enhanced Max-min Algorithm and Minimum Average Scheduling Algorithm

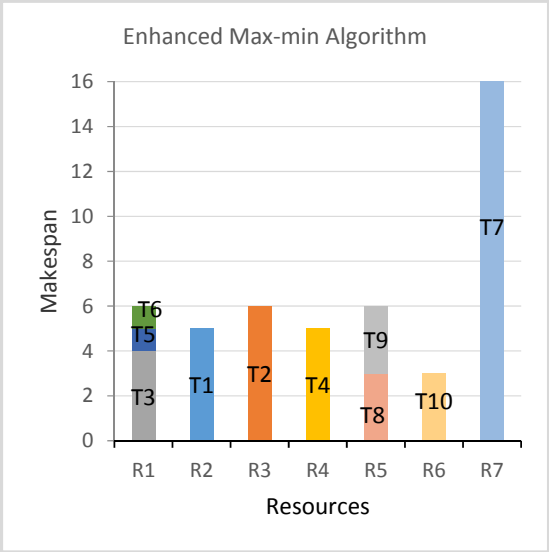
Assume that task scheduler has meta-tasks and resources with execution times for each task on each resource given below in Table 1.

Table1. Execution times of tasks

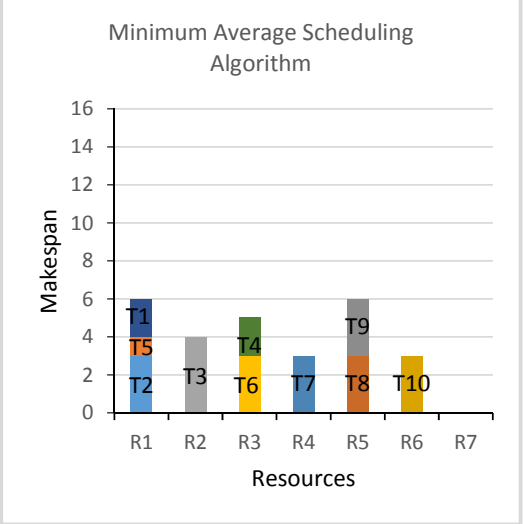
R / T	R1	R2	R3	R4	R5	R6	R7
T1	2	5	5	6	2	8	13
T2	3	2	6	7	9	9	17
T3	4	4	4	8	8	10	12
T4	5	6	2	5	7	9	18
T5	1	2	1	4	6	7	19
T6	1	1	3	8	10	6	12
T7	2	3	2	3	10	8	16
T8	5	7	4	9	3	10	13
T9	4	9	8	10	3	10	20
T10	3	11	5	10	2	3	20

Minimum Average Scheduling algorithm, MASA, Flowchart





(a)
Figure 1: (a) Gantt chart of Enhanced Max –min algorithm



(b)
Figure 1: (b) Gantt Chart of Minimum Average Scheduling Algorithm

Figure 1 describes the makespan using both the Enhanced Max–min algorithm and the MASA algorithm.
The Enhanced Max–min algorithm produces a makespan=16 ms while MASA produces a makespan=6 ms.
The results show that MASA produces a makespan smaller than that of the Enhanced Max-min algorithm.

5. Experimental Data

A simulation is made for the MASA algorithm and the Enhanced Max-min algorithm to analyze the performance of each algorithm. The simulation is written using the C++ language. The model consists of m resources, n tasks with m varying from 50 to 400 and n varying from 300 to 2700. The values of execution times of tasks are chosen randomly.
To present the advantages of the MASA algorithm against the Enhanced Max-min algorithm; Figure 2 plots the makespan versus the number of tasks for both the Enhanced Max-min algorithm and the MASA algorithm. Figure 3 plots the makespan versus the number of resources for both the Enhanced Max-min algorithm and the MASA algorithm.

In Figure 2, the number of tasks is varying from 300 to 2700 and the number of resources is constant and equals to 100. By taking 1500 tasks, the Enhanced Max-min algorithm gives makespan= 445 ms while MASA algorithm gives makespan= 294 ms. By taking 2700 tasks the makespan for Enhanced Max-min algorithm is 1106 ms while the makespan for MASA algorithm is 781 ms.
In Figure 3, the number of tasks is constant and equals to 1000 while the number of resources varies from 50 to 400. It is observed the makespan results of MASA algorithm is varying between 150 ms to 49 ms whereas the makespan results of the Enhanced Max-min algorithm is varying from 205 ms to 195 ms.

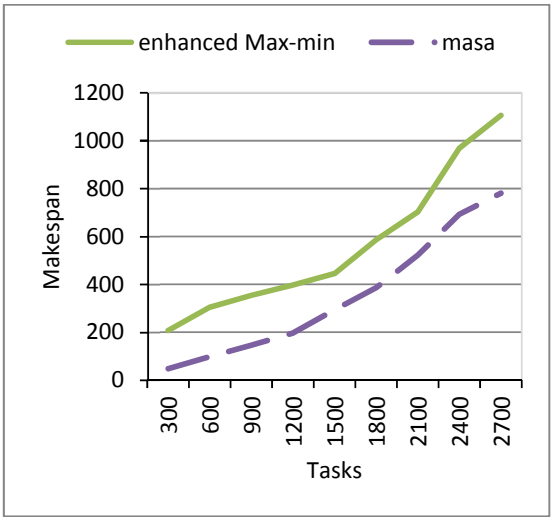


Figure 2:The makespan versus the number of tasks for both Enhanced Max-min algorithm and Minimum Scheduling Algorithm

6. Results analysis

MASA algorithm selects minimum averages tasks and assigns them to the corresponding resources. The number of selected tasks depends on the number or resources. Testing different fractions m/2, m/3… m/10 showed that the fraction m/7 gives the best result.
Suppose that the number of tasks=1000, the number of resources=100, the Enhanced Max-min algorithm produces makespan=204 ms, in MASA by applying different fractions the results are showing in table 2.

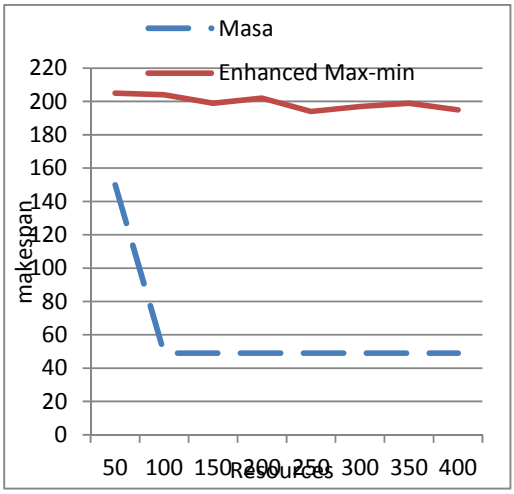


Figure 3: The makespan versus the number of resources for both Enhanced Max-min algorithm and MASA algorithm

Table2. The makespan of different fractions

fraction	makespan
m/2	71
m/3	61
m/4	59
m/5	52
m/6	50
m/7	49
m/8	49
m/9	49
m/10	49

By choosing m/2 tasks, MASA produces a makespan= 71 ms, m/3 tasks produces a makespan= 61 ms, m/4 tasks produces a makespan= 59 ms, m/5, m/6, m/7 tasks produces 52, 50, 49 ms.
It is noted that the makespan decreases as the fraction decreases from m/2 to m/7. Makespan is decreased when the number of selected tasks decreases, but at the range between m/7,...m/10 the makespan remains stable and results do not change almost.

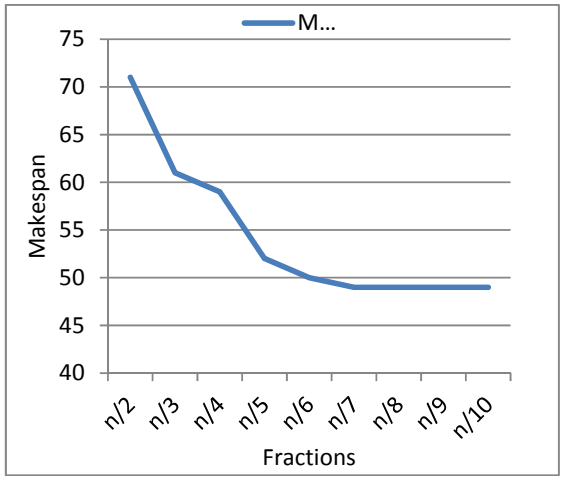


Figure4: The makespan of MASA for different fractions

It is noted from the figure that makespan decreases and then become stable. Hence, there is a breakpoint between the two fractions m/6, m/7. Since the number of tasks is an integer number, so ⌐m/7¬ is selected.

7. Conclusions

The Enhanced Max-min algorithm assigns the task with average execution time (average or nearest greater than average) to the slowest resource produces minimum completion time. When all averages are small compared to some average, the makespan increases. This paper proposed the MASA algorithm which selects ⌊m/7⌋ tasks with minimum averages execution times or nearest greater than average and assigns them to the corresponding resources producing minimum completion time. Comparing the makespan produced by the two algorithms; the results show that the MASA algorithm improves the makespan.

Future Work

The proposed algorithm selects ⌊m/7⌋ tasks with minimum averages execution times or nearest greater than average and assigns them to the corresponding resources producing minimum completion time, it may be applied in parallel computing, cloud computing, distributed systems, operating systems.

8. References

[1] Naglaa M. Reda, A. Tawfik, Mohamed A.Marzok, Soheir M. Khamis, "Sort-Mid tasks scheduling algorithm in grid computing", Journal of Advanced Research, Vol. 6(6), pp. .987–993, 2015.

[2] A. Chandak, B. Sahoo, A. Turuk, "An overview of task scheduling and performance metrics in grid computing", International Journal of Research and Reviews in Computer Science, Vol. 2(2), pp. 30–33 2011.

[3] Braun TD, Siegel HJ, Beck N, Boloni LL, Maheswaran M, Reuther AL, et al, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed

computing systems", J Parallel Distrib Compute, Vol. 61(6), pp.810–37, 2011.

[4] Elzeki OM, Rashad MZ, Elsoud MA, "Overview of scheduling tasks in distributed Computing systems ", Int J Soft Comput Eng, Vol. 2(3), pp.470–475, 2012.

[5] Neetu Goel, R.B. Garg, "A Comparative Study of CPU Scheduling Algorithms", International Journal of Graphics & Image Processing, Vol. 2(4), pp. 245-251, 2012.

[6] El-Sayed T. El-kenawy, Ali Ibraheem El Desoky, Mohamed F. Al-rahamawy, "Extended Max-Min Scheduling Using Petri Net and Load Balancing", International Journal of Soft Computing and Engineering (IJSCE), Vol. 2(4), pp. 198-203, 2012.

[7] Pinal Salot, "A Survey of various scheduling algorithm in cloud computing environment", IJRET - International Journal of Research in Engineering and Technology, Vol. 2(2), pp.131-135, 2013.

[8] Saeed Parsa, Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications, Vol. 3(4), pp. 91-99, 2009.

[9] D. Maruthanayagam, Dr. R. Uma Rani, "Enhanced Ant Colony System Based on RASA Algorithm in Grid Scheduling", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2(4), pp.1659-1674, 2011.

[10] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications, Vol. 50(12), pp.22-27, 2012.

[11] Upendra Bhoi, Purvi N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International Journal of Application or Innovation in Engineering & Management, Vol. 2(4), pp. 259-264, 2013.

Biographies

Prof. Kamal Abdelraouf ElDahshan He is a professor of Computer Science and Information Systems at Al - Azhar University in Cairo, Egypt. An Egyptian national and graduate of Cairo University, he obtained his doctoral degree from the Universitè de Technologie de Compiègne in France, where he also taught for several years. During his extended stay in France, he also worked at the prestigious Institute National de Tèlècommunications in Paris. Professor ElDahshanâAZs extensive international research, teaching, and consulting experiences have spanned four continents and include academic institutions as well as government and private organizations. He taught at Virginia Tech as a visiting professor; he was a Consultant to the Egyptian Cabinet Information and Decision Support Centre (IDSC); and he was a senior advisor to the Ministry of Education and Deputy Director of the National Technology Development Centre. Professor ElDahshan is a professional Fellow on Open Educational Resources as recognized by the United States Department of State and an Expert at ALECSO as recognized by the League of Arab States.

Dr. Afaf Abd El-Kader Abd EL-Hafiz is currently lecturer at University of Al-Azhar. She is doing her research work in Scheduling algorithms.

Nermeen Alaa Eldin Ghazy She received her B.Sc. from Faculty of Science, AL-Azhar University at 2011. and Ms. Ghazy is preparing M.Sc in scheduling algorithms.